CE7454 : Deep Learning for Data Science

Lecture 15: Recent Developments in Graph Network Architectures

Semester 1 2020/21

Xavier Bresson

https://twitter.com/xbresson

School of Computer Science and Engineering
Data Science and AI Research Centre
Nanyang Technological University (NTU), Singapore

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Outline

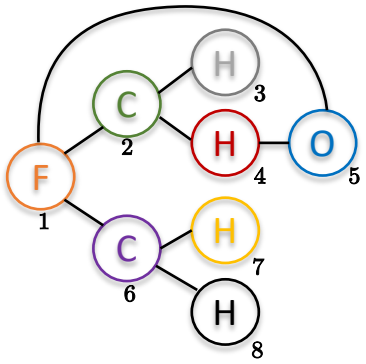
- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

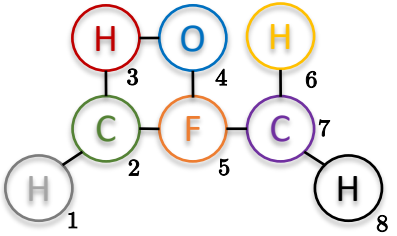
Graph Isomorphism

- Graph isomorphism : Two graphs are isomorphic if there exists an index permutation between the nodes that preserves node adjacencies.
  - An example of two isomorphic molecular graphs. Node correspondence is given by the node color.
- Determining whether two graphs are isomorphic is NP-intermediate : It is not known if a polynomial time algorithm exists, or the problem is NP-hard.
- Weisfeiler-Lehman (WL) proposed an algorithm[1] to test if two graphs are not isomorphic. However, the WL test is not sufficient to guarantee that two graphs are isomorphic.



Graph 1

Isomorphic graphs



Graph 2

[1] B Weisfeiler, A Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

WL Test[1]

- Idea : Design an injective "coloring" function f_{WL} that takes a pair (node, its neighborhood) as input, and outputs a new node color :

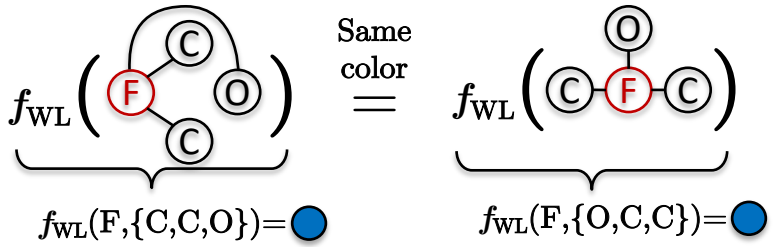
$$f_{\text{WL}}\left(c_i^{\ell}, \{c_j^{\ell}\}_{j \in \mathcal{N}_i}\right) = c_i^{\ell+1}$$

iteration

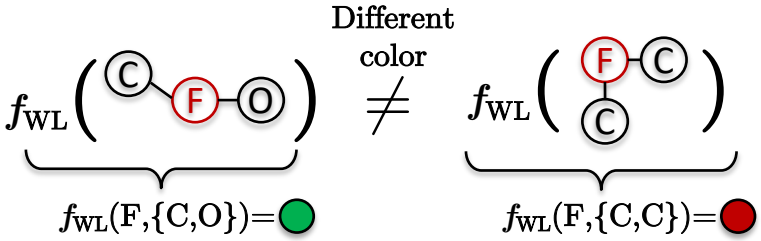
Multiset function

Multiset (set of unordered and repetitive elements)

- Properties of f_{WL} :
  - Unique color for the same pair of (node, its neighborhood).



  - Different colors for distinct pairs of (node, its neighborhood).



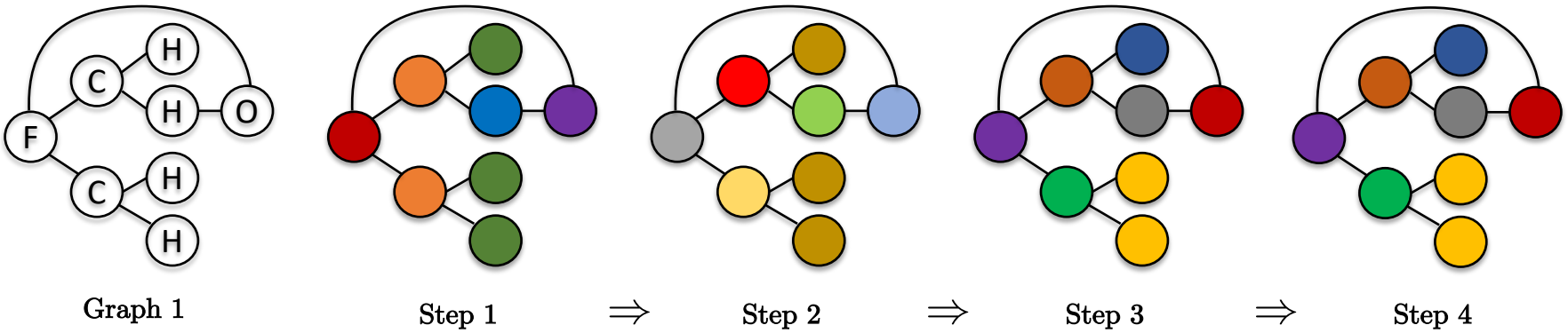
Simply said, f_{WL} must be injective (mapping different inputs to different outputs).

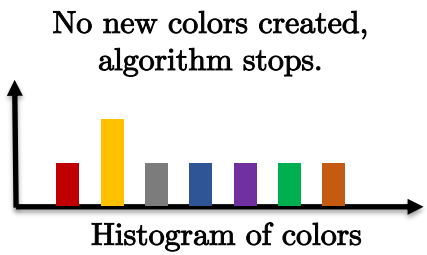
Injectivity is a discriminative property.

[1] B Weisfeiler, A Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

WL Test[1]

- WL algorithm iteratively applies the coloring function f_{WL} until no new colors are created.
- This produces a canonical representation of a graph as a histogram of colors.

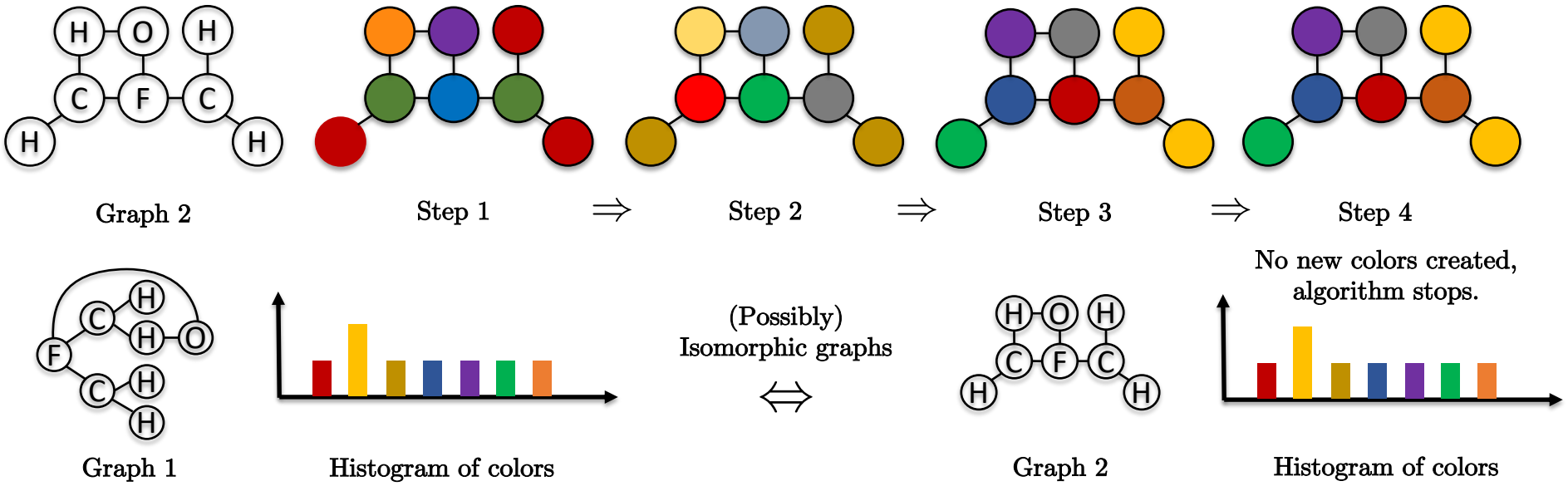




[1] B Weisfeiler, A Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

WL Test[1]

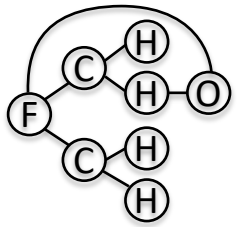
- The WL test is a necessary but not sufficient condition for isomorphism :
  - If the two graphs have different color histograms then the two graphs are guaranteed to be non-isomorphic.
  - If the two graphs produce the same histogram of colors, the graphs are possibly isomorphic (although not guaranteed).



[1] B Weisfeiler, A Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

WL-based Graph Neural Networks

- Can we turn the WL test into a neural network to classify non-isomorphic graphs ?
- Yes, but is it necessary to have a learning process for the WL test ?
  - No, but designing a constraint-free analytical injective function f_{WL} is hard and easier to do in a learning setting.
  - Distinguishing non-isomorphic graphs is not necessarily useful in practice. What is useful is the "coloring" process that can provide rich representation of data-structured graphs and can be used for downstream tasks like graph classification or regression.



$\Rightarrow$ WL-based GNNs $\Rightarrow$

Applications :
Classify non-isomorphic graphs
Graph classification/regression

Graph Isomorphism Networks (GINs)[1]

- The WL test algorithm iterates the "coloring" formula :

$$f_{\mathrm{WL}}\left(c_i^{\ell}, \{c_j^{\ell}\}_{j\in\mathcal{N}_i}\right) = c_i^{\ell+1}$$

where function f_{WL} provides :

- A unique color for the same pair $(c_i, \{c_j\}_{\mathcal{N}i})$.
- Different colors for distinct pairs $(c_i, \{c_j\}_{\mathcal{N}i})$.

This implies that f_{WL} must be injective :

$$c_i \neq c_{i'} \text{ or } \{c_j^{\ell}\}_{j\in\mathcal{N}_i} \neq \{c_j^{\ell}\}_{j\in\mathcal{N}_{i'}} \;\Rightarrow\; f_{\mathrm{WL}}\left(c_i^{\ell}, \{c_j^{\ell}\}_{j\in\mathcal{N}_i}\right) \neq f_{\mathrm{WL}}\left(c_{i'}^{\ell}, \{c_j^{\ell}\}_{j\in\mathcal{N}_{i'}}\right)$$

- Which aggregator functions are injective?

[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

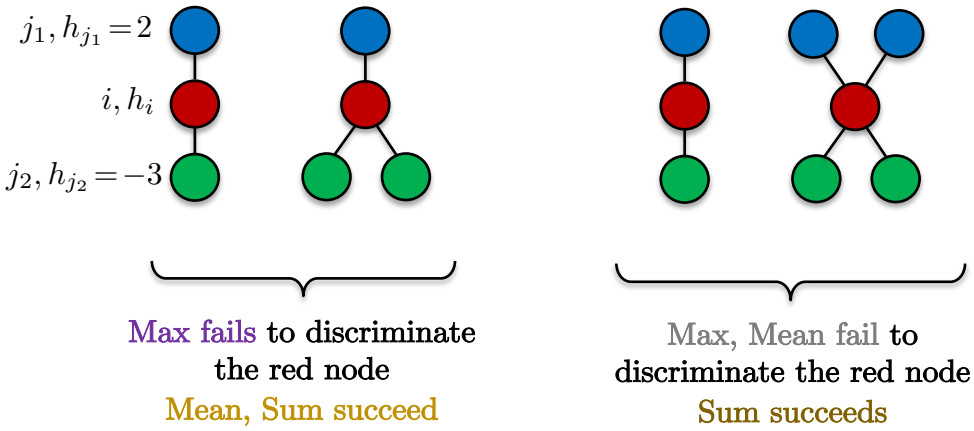
Graph Isomorphism Networks (GINs)[1]

- (Simple) aggregator functions f_{WL} :

$$h_i = \max_{j \in \mathcal{N}_i} h_j \qquad \text{Injective}$$

$$h_i = \text{mean}_{j \in \mathcal{N}_i} h_j \qquad \text{Not injective}$$

$$h_i = \text{sum}_{j \in \mathcal{N}_i} h_j \qquad \text{Not injective}$$



Injectivity is a discriminative property at the node level and by extension at the graph level.

[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

Graph Isomorphism Networks (GINs)[1]

- Can we design a NN that is maximally expressive in the sense that f_{NN} is as powerful as the WL test to distinguish non-isomorphic graphs? Yes, and such NN has an aggregation function of the form :

$$f_{\text{NN}}\left(h_i^\ell, \{h_j^\ell\}_{j\in\mathcal{N}_i}\right) = (1+\varepsilon)g(h_i^\ell) + \sum_{j\in\mathcal{N}_i} g(h_j^\ell)$$

  where we must have
  - Function g injective.
  - Sum aggregator (mean/max are not injective).
  - Constant ε must be irrational (a number that cannot be written as a ratio of two integers, π = 3.1415..)

- It is difficult to design an analytical injective function g (upper bound on the neighborhood size for countable colors, one-hot encoding does not provide meaningful distance between features).
  - A MLP can approximate g as the universal approximation theorem guarantees the existence of such function (although SGD is not guaranteed to find it):

$$h_i^{\ell+1} = f_{\text{GIN}}\left(h_i^\ell, \{h_j^\ell\}_{j\in\mathcal{N}_i}\right) = \text{MLP}^\ell\left((1+\varepsilon)h_i^\ell + \sum_{j\in\mathcal{N}_i} h_j^\ell\right)$$

Scalar ε is learned (computer cannot represent irrational numbers)

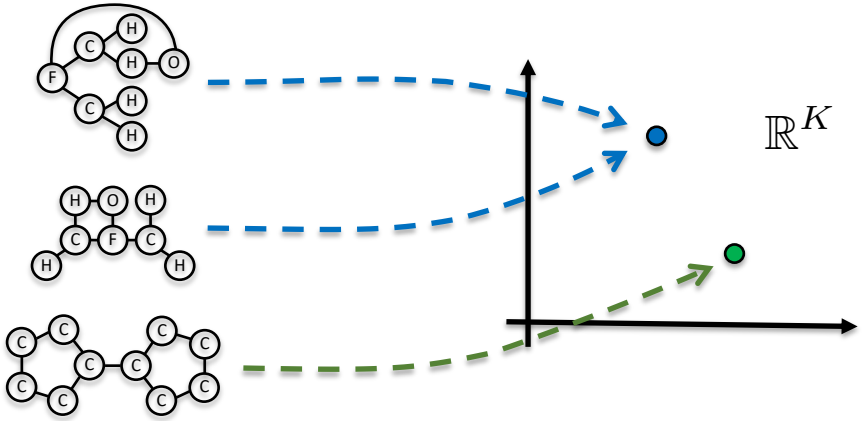
[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

Graph Isomorphism Networks (GINs)[1]

- Graph readout aggregation function must also be injective :

$$h_{\mathcal{G}} = \mathrm{MLP}\Big(\sum_{i \in V} h_i^L\Big) \in \mathbb{R}^K$$

Sum function



- Summary :
  - Injectivity : GIN can learn injective function from (graph, node features) to Euclidean space.
  - Discriminative power : GIN can learn different graph representations in R^{K} for graphs that can be distinguished by the WL test.
  - Pioneer work[1] with[2] on the theoretical question of representation/discriminative power of GNNs.

[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[2] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph networks, 2019

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

Principal Neighbourhood Aggregation[1]

- Graph Isomorphism Networks[1] introduced GNNs as expressive as the WL test to distinguish non-isomorphic graphs. Aggregation function of the form :

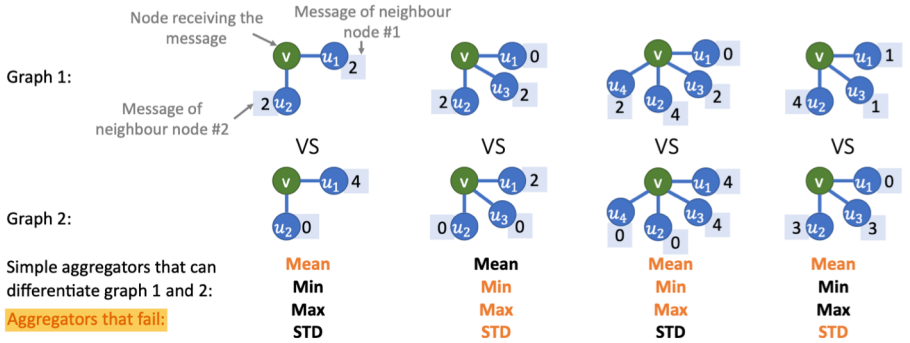
$$f_{\text{NN}}\left(h_i^\ell, \{h_j^\ell\}_{j\in\mathcal{N}_i}\right) = (1+\varepsilon)g(h_i^\ell) + \sum_{j\in\mathcal{N}_i} g(h_j^\ell)$$

  where function g is injective and sum aggregator is used (mean/max are not injective).
- Principal Neighbourhood Aggregation (PNA)[1] : Generalization of the injective sum aggregator function to multiple aggregators with degree-scalers.
  - Aggregators are mean, max, min and standard deviation.
  - Overall injective function (that guarantees GNN expressivity).
  - Increase the expressivity power of standard MP-GNNs while preserving $\mathrm{O}(n)$ complexity.

[1] Corso, Cavalleri, Beaini, Lio, and Velickovic, Principal neighbourhood aggregation for graph nets, 2020
[2] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

Principal Neighbourhood Aggregation[1]

- Illustration of aggregators are mean, max, min and standard deviation :



- PNA equation :

$$X_i^{\ell+1} = \text{Concat}\left(X_i^\ell, \tilde{X}_i^\ell \; W_1^\ell\right) W_2^\ell \in \mathbb{R}^{d_{\ell+1}}$$

$$\tilde{X}_i^\ell = \oplus_{j \in \mathcal{N}_i}(X_i^\ell, X_j^\ell) \in \mathbb{R}^{12 \times d_\ell}$$

$$= \begin{bmatrix} \text{Mean}_{j \in \mathcal{N}_i}(X_j) \\ \text{Std Dev}_{j \in \mathcal{N}_i}(X_j) \\ \max_{j \in \mathcal{N}_i}(X_j) \\ \min_{j \in \mathcal{N}_i}(X_j) \end{bmatrix} \otimes \begin{bmatrix} \text{Id} \\ S(d,1) \\ S(d,-1) \end{bmatrix}, \quad S(d,\alpha) = \left(\frac{\log(d+1)}{\delta}\right)^\alpha$$

Scalers that amplifies or decreases amplitude of incoming message.

[1] Corso, Cavalleri, Beaini, Lio, and Velickovic, Principal neighbourhood aggregation for graph nets, 2020

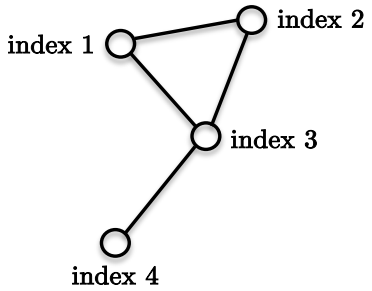
Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

Graph Representations

- Graphs are rank-*2* tensors. They can be represented as
  - List of edges (short lists for sparse graphs):

$$E = [\ \{1,2\},\ \{1,3\},\ \{2,3\},\ \{3,4\}\ ]$$



$$\mathcal{G} = (V, E)$$

  - Matrix (dense tensor):

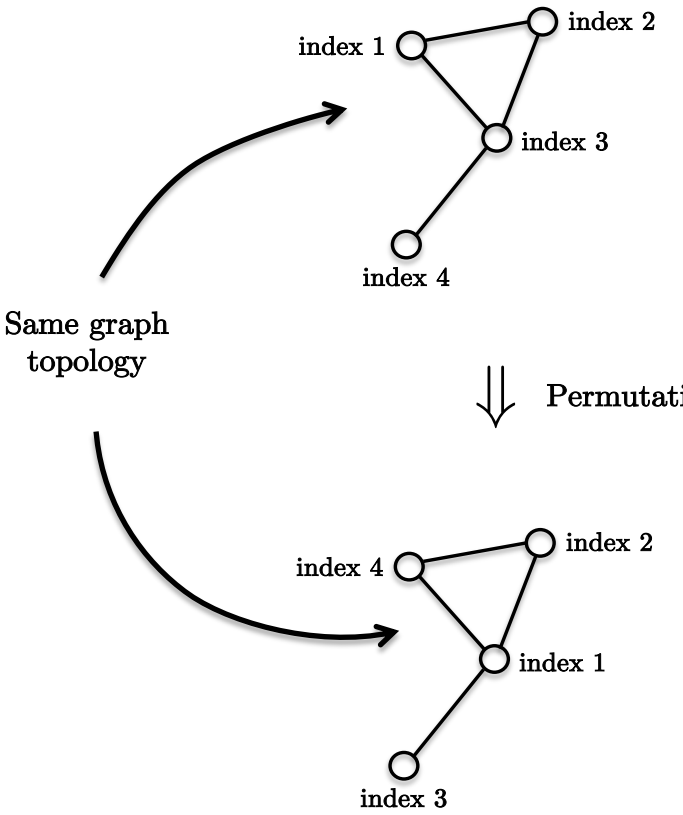
$$A = \begin{array}{c} V \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \in \mathbb{R}^{n \times n}$$

- Other tensors :
  - Vectors (ordered and unordered sets) are rank-*1* tensors : $\begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} \in \mathbb{R}^n$
  - General tensors (s.a. 3D-MRIs or hyper-graphs) of rank-m :

$\in \mathbb{R}^{n^m}$

Graph Symmetry

- The most basic graph symmetry is the invariance of the topology w.r.t. index permutation :



$$V\,[\,1\;2\;3\;4\,] \quad \begin{matrix}1\\2\\3\\4\end{matrix}\begin{bmatrix}0&1&1&0\\1&0&1&0\\1&1&0&1\\0&0&1&0\end{bmatrix} = A$$

Different adjacency matrices

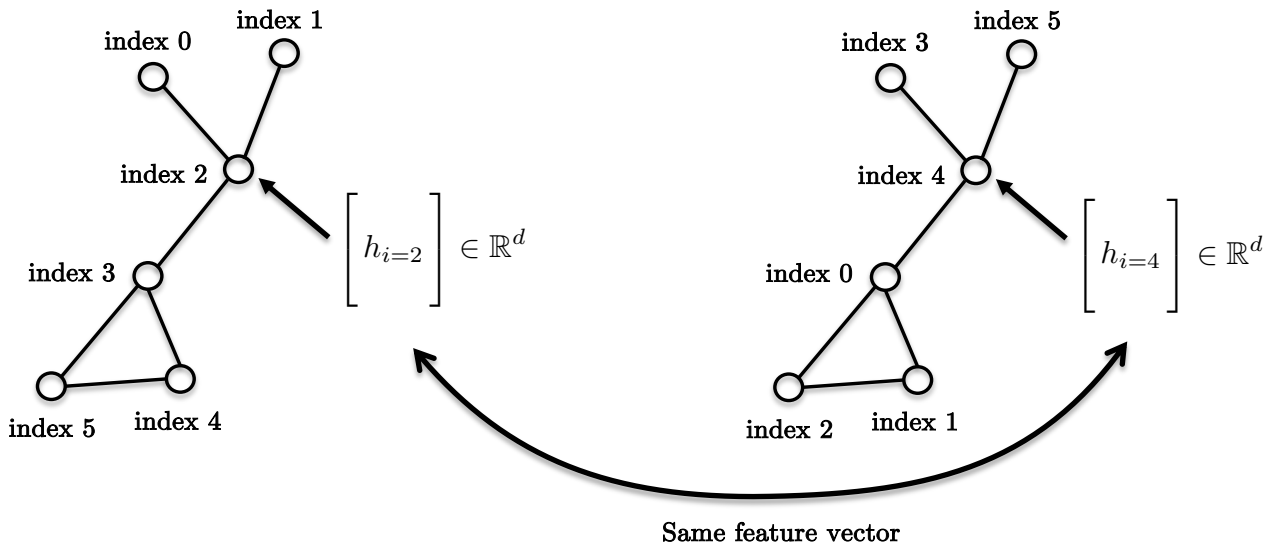
Permutation P

$$\begin{bmatrix}0&0&1&0\\0&1&0&0\\0&0&0&1\\1&0&0&0\end{bmatrix}\begin{bmatrix}0&1&1&0\\1&0&1&0\\1&1&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}0&0&0&1\\0&1&0&0\\1&0&0&0\\0&0&1&0\end{bmatrix} = P^T A P = \begin{matrix}\\1\\2\\3\\4\end{matrix}\begin{matrix}V\,[\,1\;2\;3\;4\,]\\\begin{bmatrix}0&1&1&1\\1&0&0&1\\1&0&0&0\\1&1&0&0\end{bmatrix}\end{matrix}$$

A^{new}

GNNs are Permutation Invariant

- A built-in property of GNNs is the invariance to index parametrization, i.e. node representation is independent from the choice of node ordering :



- Note that all GNNs (s.a. vanilla GCNs[1], GAT[2]) are permutation invariant.

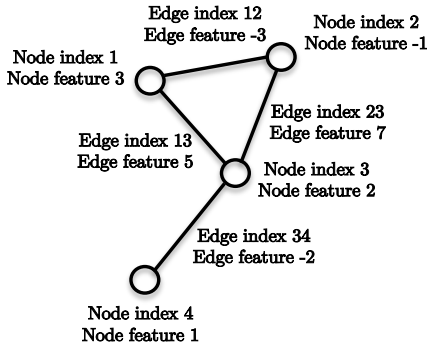
[1] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[2] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph attention networks, 2017

Equivariant Functions

- What are the functions that preserve index permutation ?
- They exist two types of such functions :
  - Equivariant functions f defined as :

$$\begin{array}{ccc} A \in \mathbb{R}^{n\times n} & \xrightarrow{f} & f(A) \in \mathbb{R}^{n\times n} \\ \downarrow P & & \downarrow P \\ P^T AP \in \mathbb{R}^{n\times n} & \xrightarrow{f} & f(P^T AP) = P^T f(A) P \in \mathbb{R}^{n\times n} \end{array}$$

  - Equivariant functions model hidden layers. For example, consider A_{ij} to be the feature of edge ij and A_{ij} to be the feature of node i :



$$A = \begin{bmatrix} 3 & -3 & 5 & 0 \\ -3 & -1 & 7 & 0 \\ 5 & 7 & 2 & -2 \\ 0 & 0 & -2 & 1 \end{bmatrix} \quad \text{and} \quad f(A) = A11^T$$

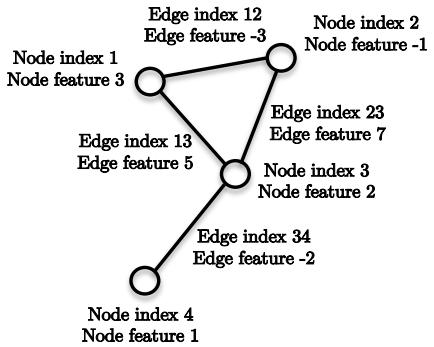
Function f satisfies : $f(P^T AP) = P^T f(A) P$

Invariant Functions

- Invariant functions g defined as :

$$\begin{array}{ccc} A \in \mathbb{R}^{n\times n} & \xrightarrow{g} & g(A) \in \mathbb{R}^K, K \geq 1 \\ \downarrow P & & \downarrow P \\ P^T AP \in \mathbb{R}^{n\times n} & \xrightarrow{g} & g(P^T AP) = g(A) \in \mathbb{R}^K, K \geq 1 \end{array}$$

- Invariant functions are readout layers. For example,



$$A = \begin{bmatrix} 3 & -3 & 5 & 0 \\ -3 & -1 & 7 & 0 \\ 5 & 7 & 2 & -2 \\ 0 & 0 & -2 & 1 \end{bmatrix} \quad \text{and} \quad g(A) = 1^T A 1 = \sum_{ij} A_{ij}$$

Function g satisfies : $g(P^T AP) = g(A)$

Symmetry & Equivariance

- The most basic image symmetry is translation.
  - Image content remains unchanged w.r.t. translations.
- What is the function that preserves translations ?
  - Convolution (translation equivariant function)
  - Convolutional neural networks[1] are translation equivariant (but not rotation equivariant).
- Generalization[2,3] :
  - Group convolution × symmetry (e.g. rotation+translation, mesh convolution)
- Why symmetry/equivariance matter ?
  - Great inductive biases
    - Less parameters to learn.
    - Faster training, better generalization.

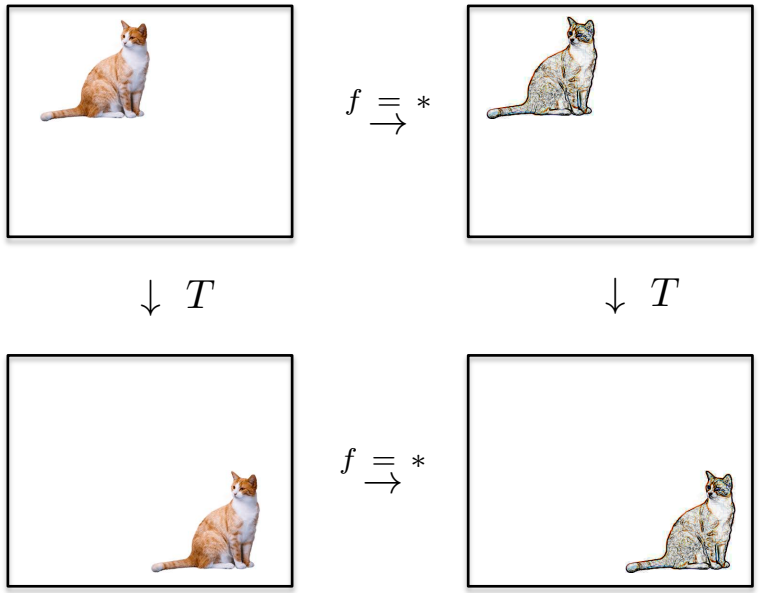


Image symmetry : Translation
Equivariant function : Convolution
(with a line detector kernel)

$$f(x) = x * k, \quad k = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

[1] LeCun, Bottou, Bengio, Haffner, Gradient-based learning applied to document recognition, 1998
[2] Cohen, Welling, Group Equivariant Convolutional Networks, 2016
[3] de Haan, Weiler, Cohen, Welling, Gauge Equivariant Mesh CNNs: Anisotropic convolutions on geometric graphs, 2020

MLP

- What is the simplest neural network that can be defined for (ordered) vectors ?
  - An MLP : Composition of { linear functions + non-linear activations/ReLU } + final linear layer for classification/regression :

$$y = m_W(x) = g_{W^L} \circ \sigma \circ f_{W^{L-1}} \circ \sigma \circ f_{W^{L-2}} \circ ... \circ \sigma \circ f_{W^0} \circ x$$

Output vector

$y \in \mathbb{R}^K, K \geq 1$

Classification/ regression layer

L linear layers

$f_W(x) = Wx + b \in \mathbb{R}^n$

Input vector

$x \in \mathbb{R}^n$

$$g_W(x) = Wx + b \in \mathbb{R}^K, K \geq 1$$

MLP for Graphs

- What is the simplest neural network that can be defined for graph adjacency matrices $A \in \mathbb{R}^{n \times n}$?
  - Equivariant GNNs (E-GNNs)[1] : Composition of { equivariant linear functions + non-linear activations/ReLU } + invariant linear function + final linear function for classification/regression :

$$y = m_{W^{L+1}} \circ g_{W^L} \circ \sigma \circ f_{W^{L-1}} \circ \sigma \circ f_{W^{L-2}} \circ ... \sigma \circ f_{W^0} \circ A$$

Output vector

$y \in \mathbb{R}^K, K \geq 1$

MLP Classification/ regression layer

$m_W(x) \in \mathbb{R}^K, K \geq 1$

Invariant linear layer

$g_W(P^T A P) = g_W(A) \in \mathbb{R}^K, K \geq 1$

L equivariant linear layers

$f_W(P^T A P) = P^T f_W(A) P \in \mathbb{R}^{n \times n}$

Input graph adjacency matrix

$A \in \mathbb{R}^{n \times n}$

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

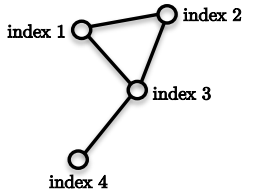
Linear Equivariant/Invariant Spaces

- Given a rank-2 tensor $A \in \mathrm{R}^{n \times n}$, can we fully characterize all equivariant/invariant linear functions $f : \mathrm{R}^{n \times n} \to \mathrm{R}^{n \times n}$ / $g : \mathrm{R}^{n \times n} \to \mathrm{R}^{K}$?
  - In other words, what is the biggest collection of equivariant/invariant linear layers ?
  - This is to achieve maximal expressivity of linear functions invariant by permutation.
- Theorem[1] : The space of all equivariant linear functions is fully characterizable and of dimension Bell(4)=15, and the space of all invariant linear functions is of dimension Bell(2)=2.
- Note that the number of functions is independent of the number of nodes n.
  - These functions can be used with datasets of graphs with variable sizes.
- General case[1] : Given an order-m tensor $A \in \mathrm{R}^{n^{\wedge} m}$, the space of all equivariant linear functions $f : \mathrm{R}^{n^{\wedge} m} \to \mathrm{R}^{n^{\wedge} m'}$ is of dimension Bell(m+m') and the space of all invariant linear functions $g : \mathrm{R}^{n^{\wedge} m} \to \mathrm{R}^{K}$ is of dimension Bell(m).

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Linear Equivariant/Invariant Functions

- The 15 equivariant linear functions $f_k : \mathrm{R}^{n \times n} \to \mathrm{R}^{n \times n}$ for a given tensor $A \in \mathrm{R}^{n \times n}$ are defined as :
  - The identity and transpose operations: $f_1(A) = A,\ f_2(A) = A^T$
  - The diag operation: $f_3(A) = \mathrm{diag}(\mathrm{diag}(A))$
  - Sum of rows replicated on rows/ columns/ diagonal:
    $$f_4(A) = A11^T,\ f_5(A) = 1(A1)^T,\ f_6(A) = \mathrm{diag}(A1),\ 1 \in \mathbb{R}^n$$
  - Sum of columns replicated on rows/ columns/ diagonal:
    $$f_7(A) = A^T11^T,\ f_8(A) = 1(A^T1)^T,\ f_9(A) = \mathrm{diag}(A^T1)$$
  - Sum of all elements replicated on all matrix/ diagonal:
    $$f_{10}(A) = (1^TA1) \cdot 11^T,\ f_{11}(A) = (1^TA1) \cdot \mathrm{diag}(1)$$
  - Sum of diagonal elements replicated on all matrix/diagonal:
    $$f_{12}(A) = (1^T\mathrm{diag}(A)) \cdot 11^T,\ f_{13}(A) = (1^T\mathrm{diag}(A)) \cdot \mathrm{diag}(1)$$
  - Replicate diagonal elements on rows/columns:
    $$f_{14}(A) = \mathrm{diag}(A)1^T,\ f_{15}(A) = 1\mathrm{diag}(A)^T$$
  - Note that some $f_k(A)$ tensors are dense, although A is sparse.



$$A = \begin{bmatrix} 0&1&1&0 \\ 1&0&1&0 \\ 1&1&0&1 \\ 0&0&1&0 \end{bmatrix}$$

$$f_1(A) = \begin{bmatrix} 0&1&1&0 \\ 1&0&1&0 \\ 1&1&0&1 \\ 0&0&1&0 \end{bmatrix} \quad f_2(A) = \begin{bmatrix} 0&1&1&0 \\ 1&0&1&0 \\ 1&1&0&1 \\ 0&0&1&0 \end{bmatrix}$$

$$f_3(A) = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix} \quad f_4(A) = \begin{bmatrix} 2&2&2&2 \\ 2&2&2&2 \\ 3&3&3&3 \\ 1&1&1&1 \end{bmatrix}$$

$$f_5(A) = \begin{bmatrix} 2&2&3&1 \\ 2&2&3&1 \\ 2&2&3&1 \\ 2&2&3&1 \end{bmatrix} \quad f_6(A) = \begin{bmatrix} 2&0&0&0 \\ 0&2&0&0 \\ 0&0&3&0 \\ 0&0&0&1 \end{bmatrix}$$

$$f_7(A) = \begin{bmatrix} 2&2&2&2 \\ 2&2&2&2 \\ 3&3&3&3 \\ 1&1&1&1 \end{bmatrix} \quad f_8(A) = \begin{bmatrix} 2&2&3&1 \\ 2&2&3&1 \\ 2&2&3&1 \\ 2&2&3&1 \end{bmatrix}$$

etc

Equivariant Layer Parametrization

- From *1*-dim input to *1*-dim output :

15 equivariant functions — 2 bias functions

$$A \in \mathbb{R}^{n \times n} \;\rightarrow\; f_W(A) = \sum_{k=1}^{15+2} W_k \, f_k(A) \;\in \mathbb{R}^{n \times n}$$

17 (learnable) parameters

- From *d*-dim input to *1*-dim output :

$$A \in \mathbb{R}^{n \times n \times d} \;\rightarrow\; f_W(A) = \sum_{k=1}^{15+2} \sum_{q=1}^{d} W_{k,q} \, f_k(A_{\cdot,\cdot,q}) \;\in \mathbb{R}^{n \times n}$$

17 x *d* (learnable) parameters

Sliced tensor *A* along the third dimension.

- From *d*-dim input to *d'*-dim output :

$$A \in \mathbb{R}^{n \times n \times d} \;\rightarrow\; \left(f_W(A)\right)_{\cdot,\cdot,q'} = \sum_{k=1}^{15+2} \sum_{q=1}^{d} W_{k,q,q'} \, f_k(A_{\cdot,\cdot,q}) \;\in \mathbb{R}^{n \times n \times d'}$$

17 x *d* x *d'* (learnable) parameters

- Memory/speed complexity is $O(n^2)$.

Running Equivariant GNNs[1]

- Start with :

$$A \in \mathbb{R}^{n \times n \times (1+d+d_e)}$$

d_e-dim edge features

Adjacency matrix

d-dim node features

- Forward pass of E-GNNs (the most expressive linear networks for graphs) :

$$y = \underbrace{m_{W^{L+1}}}_{\text{MLP layer}} \circ \underbrace{g_{W^L}}_{\text{Invariant layer}} \circ \sigma \circ \underbrace{f_{W^{L-1}} \circ \sigma \circ f_{W^{L-2}} \circ \ldots \sigma \circ \circ f_{W^0}}_{L \text{ Equivariant layers}} \circ A$$

with $f_{W^\ell} : \mathbb{R}^{n \times n \times d^\ell} \to \mathbb{R}^{n \times n \times d^{\ell+1}}$

$g_{W^L} : \mathbb{R}^{n \times n \times d^L} \to \mathbb{R}^K$

- #Parameters/layer is : $17 \times d^\ell \times d^{\ell+1}$
- Memory/speed complexity is $O(n^2)$.
- MP-GCNs can be expressed as E-GNNs[2] :

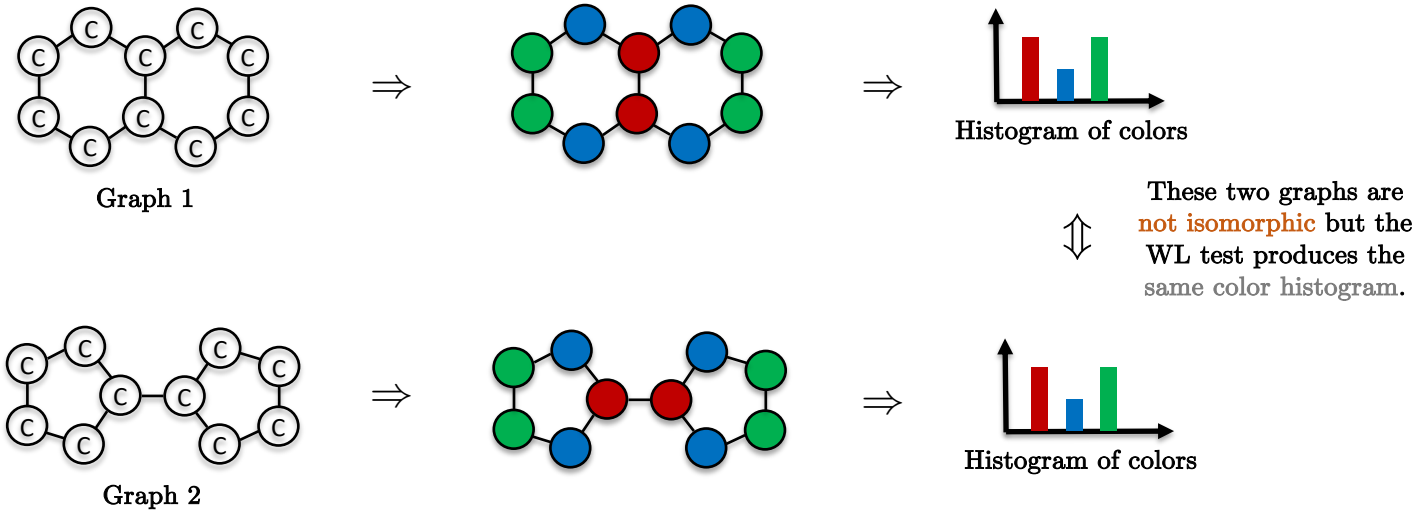
$$m_i^{\ell+1} = \sum_{j \in \mathcal{N}_i} \text{MLP}_{W^1}(h_i^\ell, h_j^\ell, e_{ij})$$

$$h_i^{\ell+1} = \text{MLP}_{W^2}(h_i^\ell, m_i^{\ell+1})$$

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[2] Gilmer, Schoenholz, Riley, Vinyals, Dahl, Neural message passing for quantum chemistry, 2017

Expressivity Power

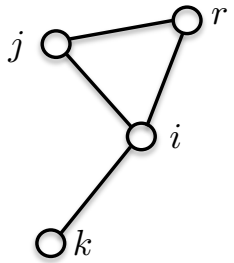
- The WL graph isomorphism test[1] can be used as a measure of expressivity/representation power of GNNs.
- GINs[2] were designed to be as maximally expressive as the original WL test[1].
- However the original WL test can fail to distinguish non-isomorphic graphs.



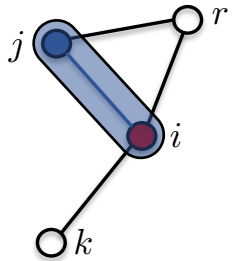
[1] B Weisfeiler, A Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968
[2] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

Expressivity Power

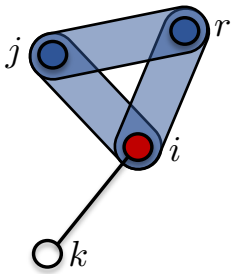
- Can we design more expressive GNNs, i.e. GNNs that are better than the original WL test[1] ?
  - k-WL tests : Original test uses 2-tuple of nodes to produce colors. To produce more colors, and improve expressivity power of WL tests, higher-order interactions between nodes with k-tuple of nodes with $k \geq 3$ can be used.



$\mathcal{G} = (V, E)$



Edges $= \{i, j\}, \{i, r\}, \{i, k\}$



Hyper-edges $= \{i, j, r\}, \{i, k, r\}, \{i, j, k\}$

2-tuple of nodes can distinguish non-isomorphic graphs with the 1-WL/2-WL tests.

3-tuple of nodes can distinguish non-isomorphic graphs with the 3-WL test.

[1] B Weisfeiler, A Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, 1968

Expressivity Power

- What is the expressivity power of equivariant GNNs in terms of WL tests ?
  - Let us define a k-order Equivariant GNNs :

$$y = m_{W^{L+1}} \circ g_{W^L} \circ \sigma \circ f_{W^{L-1}} \circ \sigma \circ f_{W^{L-2}} \circ ... \sigma \circ f_{W^0} \circ A$$
$$\text{with } k = \max_{\ell \in [0, L-1]} k_\ell$$
$$\text{and } f_{W^\ell} : \mathbb{R}^{n^{k_\ell} \times d_\ell} \to \mathbb{R}^{n^{k_{\ell+1}} \times d_{\ell+1}}$$

- Theorem[1] : There exists a k-order E-GNN that can distinguish non-isomorphic graphs with the k-WL test.
  - Existence result (although SGD is not guaranteed to find such network).
  - It is not a practical result as k-order E-GNNs requite $O(n^k)$ memory/speed complexities.
    - Note that 1-WL/2-WL GNNs (s.a. GINs) have the same discriminative power[2]. It implies that we need at least $k=3$ to be more powerful than GINs. But then, it requires $O(n^3)$.

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[2] Cai, Furer, Immerman, An optimal lower bound on the number of variables for graph identification, 1992

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

3-WL GNNs[1]

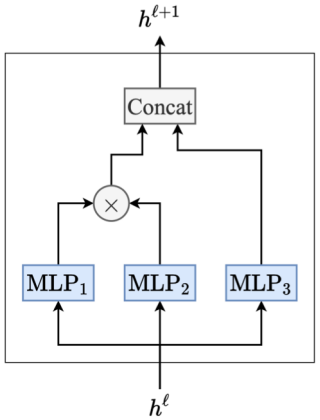
- How to design GNNs that are provable 3-WL but do not require $O(n^3)$ memory/speed complexities ?
- 3-WL GNNs[1] : To achieve higher interactions between nodes, it was proposed to multiply matrices (that generates non-local interactions).
  - Theorem : There exists a 3-WL GNN as expressive as the 3-WL test.
  - Memory is quadratic $O(n^2)$ but matrix-matrix multiplication implies $O(n^3)$ speed.
    - It improves memory complexity compared to E-GNNs[2].
    - Matrix-matric multiplication densifies sparse matrix.

[1] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[2] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

3-WL GNNs[1]

- 3-WL GNN update layer :

$$h^{\ell+1} = \text{Concat}\big(m_{W_1^\ell}(h^\ell) \cdot m_{W_2^\ell}(h^\ell),\ m_{W_3^\ell}(h^\ell)\big),$$
$$\text{where } h^\ell, h^{\ell+1} \in \mathbb{R}^{n\times n\times d}, W_a, W_b \in \mathbb{R}^{d\times d},$$



where m_W are 2-layer MLPs applied along the feature dimension. The matrix-matrix multiplication is carried out along the first and second dimensions as :

$$\big(M_{W_1}(h)\ .\ M_{W_2}(h)\big)_{i,j,k} = \sum_{p=1}^{n} \big(M_{W_1}(h)\big)_{i,p,k} \cdot \big(M_{W_2}(h)\big)_{p,j,k}$$

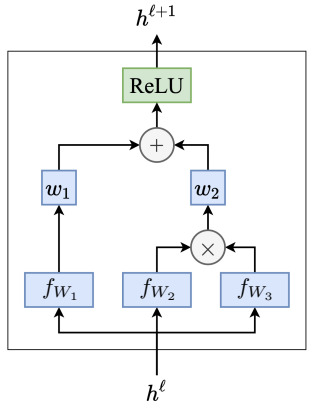
with complexity $O(n^3)$.

[1] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019

RingGNNs[1]

- A related method to *3*-WL GNNs to design more expressive GNNs than GINs.
  - Higher-order interaction between nodes is produced by multiplying equivariant linear layers.
  - RingGNN update layer :

$$h^{\ell+1} = \sigma\big(w_1^\ell \; f_{W_1^\ell}(h^\ell) + w_2^\ell \; f_{W_2^\ell}(h^\ell) \cdot f_{W_3^\ell}(h^\ell)\big),$$
$$\text{where } h^\ell, h^{\ell+1} \in \mathbb{R}^{n\times n\times d}, w_{1,2}^\ell \in \mathbb{R}, W \in \mathbb{R}^{d\times d\times 17},$$



where f_W are the equivariant linear layers defined in[2] as :

$$\big(f_W(A)\big)_{\cdot,\cdot,q'} = \sum_{k=1}^{15+2} \sum_{q=1}^{d} W_{k,q,q'} \; f_k(A_{\cdot,\cdot,q}) \;\; \in \mathbb{R}^{n\times n\times d'}$$

and f_k are the 15 equivariant linear functions $f_k : \mathrm{R}^{n \times n} \to \mathrm{R}^{n \times n}$ for a given tensor $A \in \mathrm{R}^{n \times n}$, and 2 bias functions.

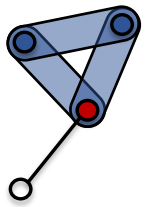
[1] Chen, Villar, Chen, Bruna, On the equivalence between graph isomorphism testing and function approximation with gnns, 2019
[2] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

Sparse k-WL-GNNs[1]

- Original k-WL GNNs are provable as expressive as the k-WL test, but they are not practical because of $O(n^k)$ memory/speed requirements.
  - Dense rank-k tensors represent all possible k-tuple of nodes (existing or not).
  - These GNNs do not leverage graph sparsity, which is known to be a good inductive bias for generalization. Also, dense tensors increase over-fitting.
- Sparse k-WL-GNNs[1] solely consider the k-tuple of nodes present in the given graphs.
  - Theorem[1] : Sparse k-WL-GNNs are strictly more powerful than dense k-WL-GNNs.
  - Theorem[1,2] : Sparse k-WL-GNNs exhibit a smaller generalization error than dense k-WL-GNNs.
  - In practice, the memory and speed complexities are improved, and benefit from the graph sparsity. However, the numerical time still remains large as all k-tuple of nodes present in the graph must be considered at each layer.



[1] Morris, Rattan, Mutzel, Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings, 2020
[2] Garg, Jegelka, Jaakkola, Generalization and representational limits of graph neural networks, 2020

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

Graph Low-Rank Global Attention[1]

- E-GNNs[2] require $O(n^3)$ memory/speed complexity, which was reduced to $O(n^2)/O(n^3)$ in[3].
- Can we further reduce the complexity while keeping the *3*-WL expressivity power ?
  - Adding a global attention module to MP-GCNs can improve the expressivity power to *2*-FWL, which is equivalent to the *3*-WL test.
  - Global attention requires $O(n^2)/O(n^3)$ in Transformers[4].
  - Low-rank global attention reduces the complexity to $O(nk)/O(nk^2)$ with a matrix of order k, with k≪n.

$$X^{\ell=0} \in \mathbb{R}^{n \times d_0}$$

$$X^{\ell} \in \mathbb{R}^{n \times d_\ell}$$

$$X^{\ell+1} = \mathrm{Concat}\big(X^\ell, A(X^\ell), \underset{n \times d_\ell}{\mathrm{GNN}}(X^\ell)\big) W \in \mathbb{R}^{n \times d_{\ell+1}}, \; W \in \mathbb{R}^{(2d_\ell + 2k) \times d_{\ell+1}}$$

Low-rank global attention

$$A(X) = \mathrm{Concat}\Big(\frac{1}{\eta(X)} \underset{n \times k}{m_1(X)} \big(\underset{k \times k}{m_2(X)^T m_3(X)}\big), \underset{n \times k}{m_4(X)}\Big) \in \mathbb{R}^{n \times 2k}$$

MLP

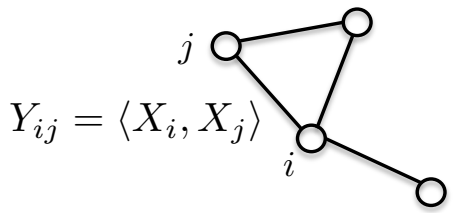
Normalizing factor

$$\eta(X) = \frac{1}{n}\big(1^T m_1(X)\big)\big(m_2(X)^T 1\big)$$

[1] Puny, Ben-Hamu, Lipman, From Graph Low-Rank Global Attention to 2-FWL Approximation, 2020
[2] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[3] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[4] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017

Graph Low-Rank Global Attention[1]

- Theorems[1] :
  - The proposed GNN model is as powerful as *3*-WL under the assumption that the graph is a rich feature graph (RFG).
  - The proposed GNN is universal, i.e. it can approximate any arbitrary continuous function over the class of RFGs.
- Def : A RFG carries all structural and feature information in their node feature.
  - Consequence 1 : RFG node features X fully characterize the edge isomorphism type.
    - Two pairs (i,j) and (i',j') have the same isomorphism type iff $X_i=X_{i'}$ and $X_j=X_{j'}$.
  - Consequence 2 : A RGB is a graph s.t. the isomorphism type Y_{ij} of the pair (i,j) and the node features X can be expressed as $Y=XX^T$.
  - Consequence 3 : Given a graph adjacency matrix A and a unique node representation c, then the graph is a RFG and $Y=(A,c1^T,1c^T)$ represents the isomorphism type of the graph.
- Observation : Most real-world graphs with various node features are almost RGBs because the diversity of features may provide almost a unique representation.



[1] Puny, Ben-Hamu, Lipman, From Graph Low-Rank Global Attention to 2-FWL Approximation, 2020

Graph Low-Rank Global Attention[1]

- There is an algorithmic alignment between the 2-FWL algorithm and the proposed GNN for RFGs :

$$
\begin{aligned}
Y^{\ell+1} &= \mathrm{Concat}\big(Y^{\ell}, \mathrm{ENC}(Z^{\ell})\big) && \text{Multiset encoding of neighbor colors} \\
\mathrm{ENC}(Z) &= \{Y^{\beta}Y^{\gamma} \mid |\beta| + |\gamma| \leq n\} && \text{Encoding with matrix multiplication of monomials} \\
&= Y^{\beta}Y^{\gamma} && \text{Select one element for practical reason} \\
&= \Psi_{\beta}(X)\Phi_{\beta}(X)^{T}\Psi_{\gamma}(X)\Phi_{\gamma}(X)^{T} && \text{Represent the monomial product with polynomial kernels} \\
&= X^{\ell+1}X^{\ell+1^{T}} && \text{Represent Y as a RFG} \\
X^{\ell+1} &= \mathrm{Concat}\big(X^{\ell}, \Psi_{\beta}(X)\Phi_{\beta}(X)^{T}\Psi_{\gamma}(X), \Phi_{\gamma}(X)\big) \\
X^{\ell+1} &= \mathrm{Concat}\big(X^{\ell}, \frac{1}{\eta(X)}m_1(X)\big(m_2(X)^{T}m_3(X)\big), m_4(X)\big)
\end{aligned}
\quad \Big\} \text{ Algorithmic alignment}
$$

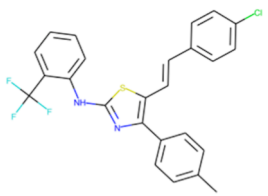
[1] Puny, Ben-Hamu, Lipman, From Graph Low-Rank Global Attention to 2-FWL Approximation, 2020

Outline

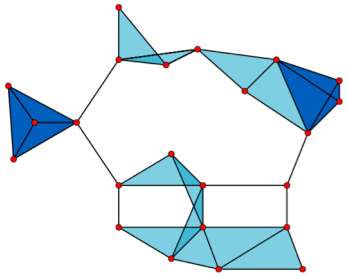
- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

Graph Substructure Networks[1]

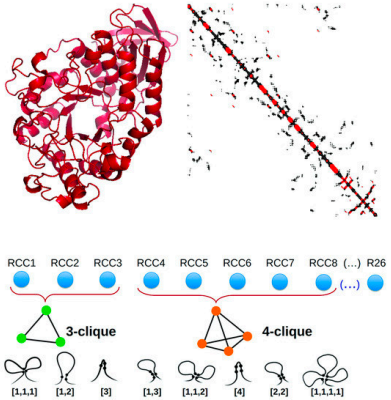
- Dense or sparse k-WL GNNs require to use k-tuple of nodes.
- Some k-tuple of nodes are informative and representative of specific classes of graphs.
  - Cycles are useful in chemistry (molecular rings).
  - Cliques are present in social networks (communities) and protein-protein interactions (PPI) in biological networks.



ZINC molecule
with 3 rings



Graph with
23 × 1-vertex cliques (vertices)
42 × 2-vertex cliques (edges)
19 × 3-vertex cliques (light and dark blue triangles)
and 2 × 4-vertex cliques (dark blue areas)



Residue Cluster class (RCC)
construction[2]

[1] Bouritsas, Frasca, Zafeiriou, Bronstein, Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting, 2020
[2] Velez, Fontove, Del Rio, Protein–Protein Interactions Efficiently Modeled by Residue Cluster Classes, 2020

Graph Substructure Networks[1]

- Note that cycles, cliques, paths cannot be identified with standard message-passing GNNs[2,3].
  - *1*-WL/*2*-WL GNNs cannot identify substructures consisting of 3 or more nodes.
  - Only *k*-WL GNNs can find substructures consisting of k or less nodes, but they are computationally expensive with $O(n^k)$ complexity.
- Graph Substructure Networks[1] :
  - Augment MP-GNNs with subgraph isomorphism counts.
  - Subgraph isomorphism extract node/edge structural features s.a. ring or click counts.
  - Counting is done as a pre-processing step.

$$m_i^{\ell+1} = \sum_{j \in \mathcal{N}_i} \text{MLP}_{W^1}(h_i^\ell, h_j^\ell, e_{ij})$$
$$h_i^{\ell+1} = \text{MLP}_{W^2}(h_i^\ell, m_i^{\ell+1})$$

Standard MP-GNNs[4]

$\Rightarrow$

$$m_i^{\ell+1} = \sum_{j \in \mathcal{N}_i} \text{MLP}_{W^1}(h_i^\ell, h_j^\ell, e_{ij}, x_i^{\mathcal{S}}, x_j^{\mathcal{S}}, x_{ij}^{\mathcal{S}})$$
$$h_i^{\ell+1} = \text{MLP}_{W^2}(h_i^\ell, m_i^{\ell+1})$$

MP-GNNs augmented with subgraph isomorphism counts

[1] Bouritsas, Frasca, Zafeiriou, Bronstein, Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting, 2020
[2] Arvind, Fuhlbruck, Kobler, Verbitsky, On Weisfeiler-Leman invariance: subgraph counts and related graph properties, 2020
[3] Chen, Chen, Villar, Bruna, Can graph neural networks count substructures? 2020
[4] Gilmer, Schoenholz, Riley, Vinyals, Dahl, Neural message passing for quantum chemistry, 2017

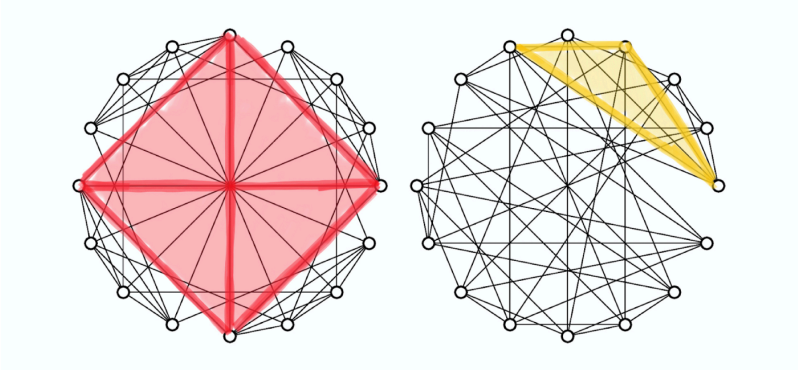
Graph Substructure Networks[1]

- Properties :
  - Combine message passing-GNNs with network motifs and graphlet techniques[2,3,4].
  - Complexity is linear/O(n) in terms of memory and space for sparse graphs.
  - Subgraph isomorphic counting requires O(n^k) by examining all possible k-tuples in the graph, but this pre-processing step is done once.
  - Subgraph structures are hand-crafted, and required prior domain knowledge. Can we learn the most informative substructure ?
  - Universality : If the Reconstruction Conjecture[5] holds, GSN can distinguish all non-isomorphic graphs when using substructures of size $k{=}n{-}1$.

[1] Bouritsas, Frasca, Zafeiriou, Bronstein, Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting, 2020
[2] Milo, Shen-Orr, Itzkovitz, Kashtan, Chklovskii, Alon, Network motifs: simple building blocks of complex networks, 2002
[3] Przul, Biological network comparison using graphlet degree distribution. Bioinformatics, 2007
[4] Shervashidze, Vishwanathan, Petri, Mehlhorn, Borgwardt, Efficient graphlet kernels for large graph comparison, 2009
[5] Kelly, A congruence theorem for trees, 1957

Graph Substructure Networks[1]

- Properties :
  - Strictly more powerful than *2*-WL GNNs with substructures consisting of $k \geq 2$ nodes.
  - Can be at least more powerful than *3*-WL GNNs for regular graphs :



Rook and Shrikhande non-isomorphic graphs

These 2 graphs cannot be distinguished by the *3*-WL test as all nodes see the same *3*-tuple.

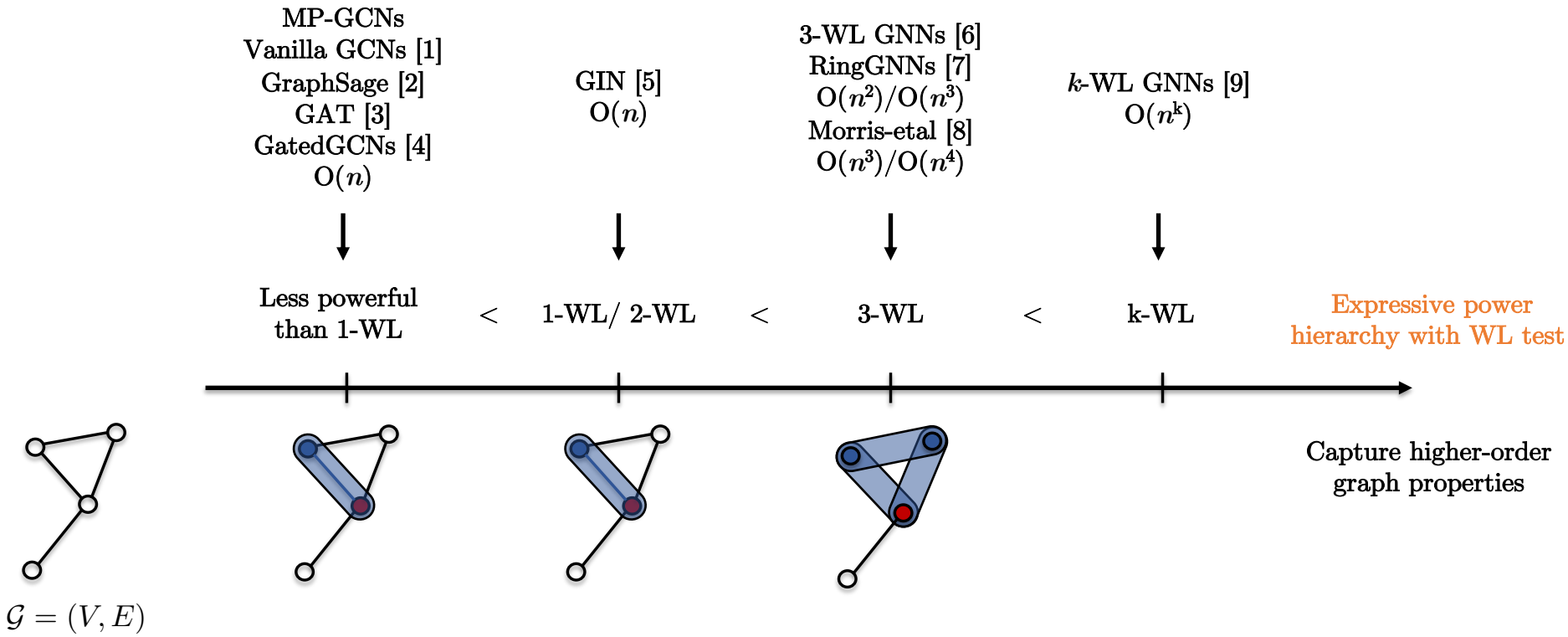
Higher-order tuples are required s.a. 4-clique for the left graph and 3-clique for the right one.

[1] Bouritsas, Frasca, Zafeiriou, Bronstein, Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting, 2020

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- **Expressivity Power and Universal Approximator**
  - **Expressivity with WL**
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

Expressivity Power with WL Test



[1] Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2017
[2] Hamilton, Ying, Leskovec, Inductive representation learning on large graphs, 2017
[3] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph attention networks, 2017
[4] Bresson, Laurent, Residual gated graph convnets, 2017
[5] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[6] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[7] Chen, Villar, Chen, Bruna, On the equivalence between graph isomorphism testing and function approximation with gnns, 2019
[8] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, 2019
[9] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- **Expressivity Power and Universal Approximator**
  - Expressivity with WL
  - **Graph Universal Approximator**
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

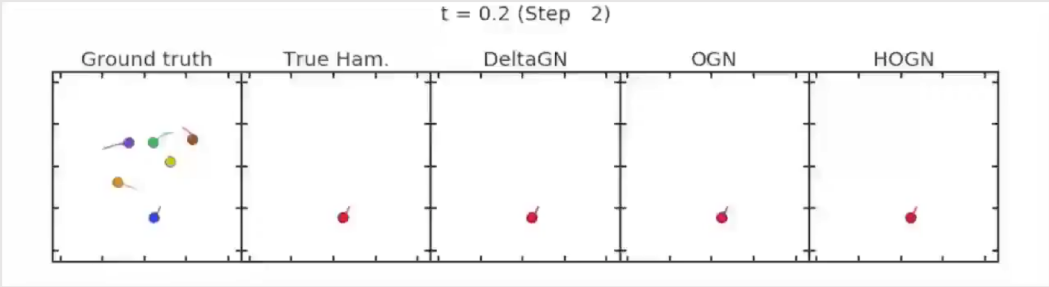
Universal Approximator

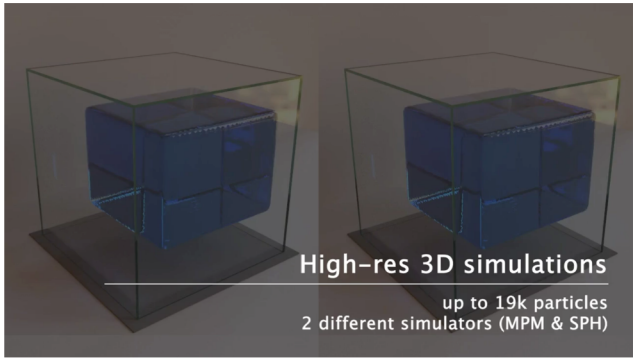
- What functions can be approximated by MLP ?
  - Universal approximation theorem[1,2] : Any continuous function can be arbitrarily approximated by a MLP, with the necessary condition that the number of hidden neurons goes to infinity (or very large).
- What functions can be approximated by Equivariant GNNs ?
  - Theorem[3] : Any continuous function invariant by permutation can be arbitrarily approximated by E-GNNs, with the necessary condition the network has a tensor of order m=poly(n)=n(n-1)/2, where n is the number of nodes.
  - High-order tensors are not practical as memory/speed complexities grow as $O(n^m)$.
- Result in [3] is for the worst case, and was improved in [4] to be independent of n (but no upper bound on the order is known).

[1] Cybenko, Approximation by superpositions of a sigmoidal function, 1989
[2] Hornik, Approximation capabilities of multilayer feedforward networks, 1991
[3] Maron Fetaya, Segol, Lipman, On the universality of invariant networks, 2019
[4] Keriven, Peyré, Universal invariant and equivariant graph neural networks, 2019

Universal Approximator and Generalization

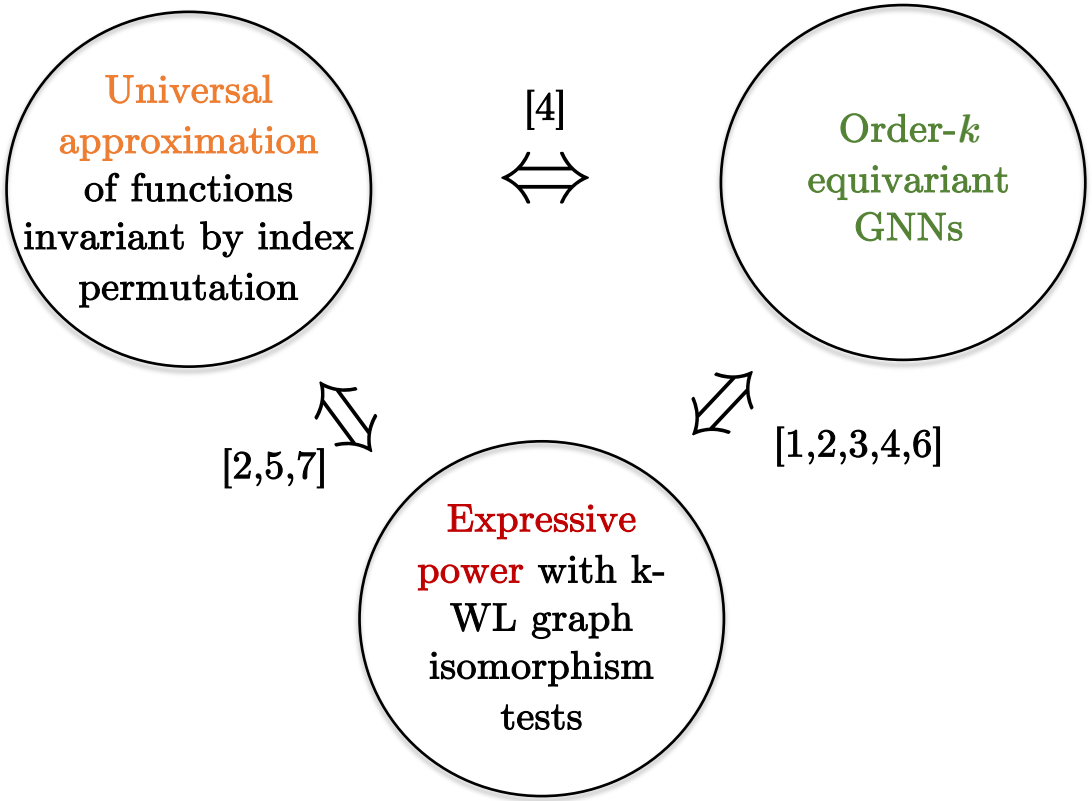
- Universal approximator for functions defined on graphs and expressiveness/representation power to distinguish non-isomorphic graphs are necessary conditions to design good GNNs.
- However, these properties do not guarantee strong generalization performances. It is important to make the distinction between representation power and generalization.
  - MLP are universal approximator for general functions but they do not generalize well.
  - Equivariant GNNs are MLP for permutation-invariant functions but they do not generalize as well as GAT/GatedGCNs[1].
- GNNs that are universal approximators and that benefit from specific inductive biases can lead to strong generalization performances.
  - For example, combining GNNs with physical constraints like Hamiltonian mechanics[2,3].





[1] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[2] Sanchez-Gonzalez, Bapst, Cranmer, Battaglia, Hamiltonian Graph Networks with ODE Integrators, 2019
[3] Sanchez-Gonzalez, Godwin, Pfaff, Ying, Leskovec, Battaglia, Learning to simulate complex physics with graph networks, 2020

Relationship Between Properties



[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[2] Maron Fetaya, Segol, Lipman, On the universality of invariant networks, 2019
[3] Keriven, Peyré, Universal invariant and equivariant graph neural networks, 2019
[4] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019
[5] Chen, Villar, Chen, Bruna, On the equivalence between graph isomorphism testing and function approximation with gnns, 2019
[6] Morris, Ritzert, Fey, Hamilton, Lenssen, Rattan, Grohe, Weisfeiler and leman go neural: Higher-order graph neural networks, 2019
[7] Loukas, What graph neural networks cannot learn: depth vs width, 2019

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- **Expressivity Power and Universal Approximator**
  - Expressivity with WL
  - Graph Universal Approximator
  - **Limitations of Expressivity**
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

What GNNs Cannot Learn[1]

- Study of expressive power of Message Passing GNNs[2] with node identifier :

$$m_{j\to i}^{\ell} = \mathrm{MLP}_{\mathrm{message}}^{\ell}(h_i^{\ell}, h_j^{\ell}, \mathrm{id}_i^{\ell}, \mathrm{id}_j^{\ell}, e_{ij}) \quad \text{Message Passing}$$
$$h_i^{\ell} = \mathrm{MLP}_{\mathrm{update}}^{\ell}\Big(\sum_{j\to i} h_j^{\ell}\Big) \quad \text{Update}$$

- Sufficient conditions for Turing universal[2] : Message passing and Update layers are Turing-complete (guaranteed by MLPs), depth $d \geq \delta_G$ layers (graph diameter), width is unbounded, and each node is uniquely identified.
- As Turing universality is strictly stronger than universal approximation[3], then Turing universal GNNs can solve the graph isomorphism problem.
  - Note that universality/expressive power does not reveal much on generalization performance and SGD is not guaranteed to find the universal GNNs.

[1] Loukas, What graph neural networks cannot learn: depth vs width, 2019
[2] Angluin, Local and global properties in networks of processors, 1980
[3] Chen, Villar, Chen, Bruna, On the equivalence between graph isomorphism testing and function approximation with gnns, 2019

What GNNs Cannot Learn[1]

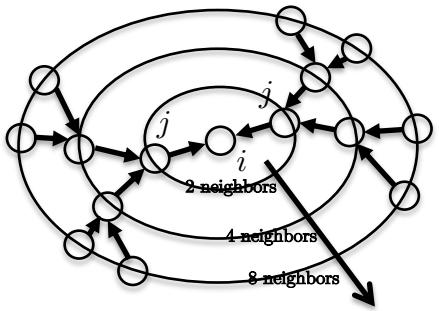
- Impossibility results : GNNs cannot compute (and consequently learn) some graph-theoretical problems if the network capacity (product of depth and width) is limited, i.e. capacity $\leq O(n^{\delta})$ for δ in [½,2].
  - Decision problems : Subgraph recognition, detecting cycles, subgraph is connected, forms a spanning tree, is bipartite, is a cut, is an s-t cut, is a Hamiltonian cycle or a simple path.
  - Optimization problems : Polynomial-time problems (minimum cut, shortest s-t path, minimum spanning tree) and NP-hard problems (minimum vertex cover, maximum independent set, coloring).
  - Estimation problems : Computing graph diameter and girth (shortest cycle contained in graph).
- It is an alternate approach to study expressive power of GNNs, different from WL graph isomorphic tests.
- Proposed analysis of GNN efficiency is worst-case (average-case is missing for practical applications).

[1] Loukas, What graph neural networks cannot learn: depth vs width, 2019

On the Bottleneck of Graph Neural Networks[1]

- Aggregation functions in MP-GNNs suffers from a bottleneck problem :
  - Observe that the number of nodes in the receptive field grows exponentially with the number of layers.
  - This causes over-squashing : information from the exponentially-growing receptive field is compressed into fixed-length vectors (by aggregation).
  - Bad consequence for deep GNNs and graph problems that require long-range node interaction (with as many layers as the desired radius of the receptive field).
  - GNNs can fail to propagate long-distance information (causing overfitting and poor generalization).

$$m_{j\to i}^{\ell} = \mathrm{MLP}_{\mathrm{message}}^{\ell}(h_i^{\ell}, h_j^{\ell}) \quad \text{Message Passing}$$
$$h_i^{\ell+1} = \mathrm{MLP}_{\mathrm{update}}^{\ell}\Big(\sum_{j\to i} h_j^{\ell}\Big) \quad \text{Aggregate}$$



Nb of nodes in the reception field grows exponentially with nb of layers.

[1] Alon, Yahav, On the Bottleneck of Graph Neural Networks and its Practical Implications

On the Bottleneck of Graph Neural Networks[1]

- Proposed solution :
  - A fully connected graph layer (FCG) layer (every pair of nodes is connected by an edge) :

$$m_{ji}^{\ell} = \mathrm{MLP}_{\mathrm{message}}^{\ell}(h_i^{\ell}, h_j^{\ell}) \quad \text{Message Passing}$$
$$h_i^{\ell+1} = \mathrm{MLP}_{\mathrm{update}}^{\ell}\Big(\sum_{j \in V} m_{ij}^{\ell}\Big) \quad \text{Aggregate}$$

  - The FCG layer is simply added to existing GNN architectures, after the GNN layers and before the task-based readout layer.
  - Complexity is $O(n^2)$.
- It improves performances without any hyperparameter tuning or additional weights.
- Architecture with all layers as FCG layers (graph structure is ignored) perform badly.
- Over-squashing is more problematic than under-reaching : Even when information is reachable within L edges (using L layers), this information might fail to flow in the bottleneck.

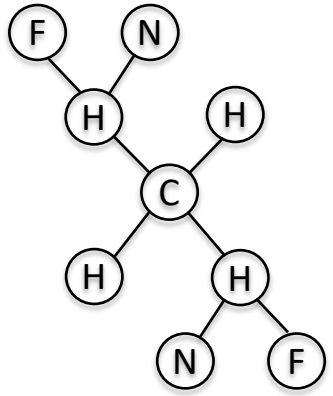
[1] Alon, Yahav, On the Bottleneck of Graph Neural Networks and its Practical Implications

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- **Graph Positional Encodings**
  - **Index Positional Encodings**
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

Structural (GNNs) vs Positional Encodings

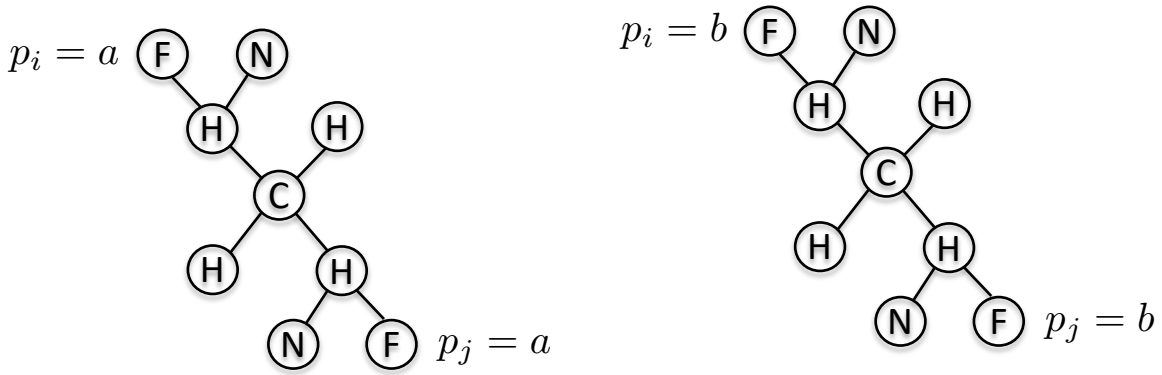
- GNNs like GINs[1] are not able to differentiate representations of isomorphic nodes, i.e. nodes with the same neighborhood structure :
  - All F atoms have the same representation.
  - All N atoms have the same representation.
  - Two H atoms have the same representation.
  - Two H atoms have the same representation.
- This is a limitation of the expressivity of GNNs.
  - They are not designed to differentiate isomorphic nodes.
  - Can we break this structural symmetry ?
  - Yes, this can be done either by
    - Higher-order WL-GNNs, but they are not practical $O(n^k)$.
    - Positional encodings of nodes.



[1] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

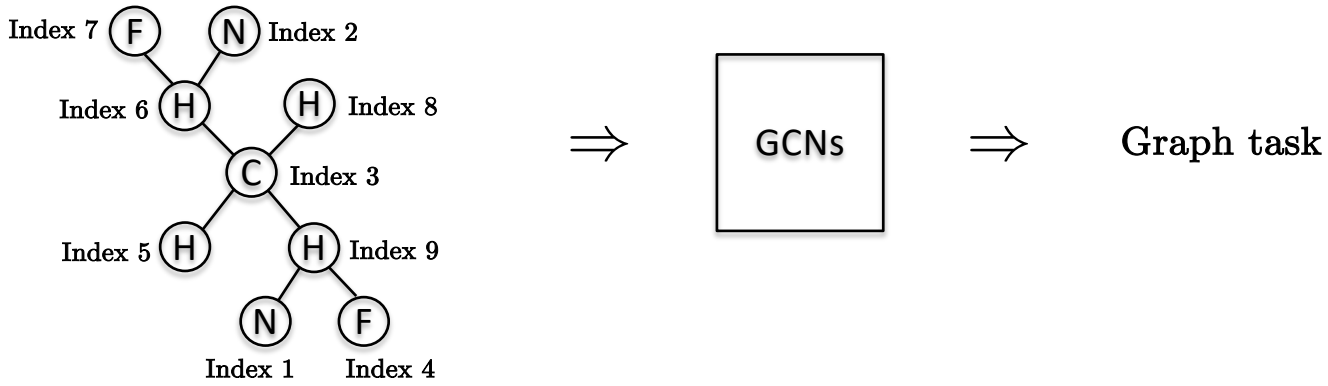
Graph Positional Encodings

- Properties of PEs :
  - Unique representation for each node.
  - Distance-sensitive : Nodes far apart on the graph should have different positional features whereas nodes nearby have similar positional features.
- Graph symmetries prevent assigning canonical representation of PEs.
  - if nodes i and j are structurally symmetric and we have PEs $p_i{=}a,\ p_j{=}b$, then it is also possible to arbitrary choose $p_i{=}b,\ p_j{=}a$.
  - PEs are always arbitrary up to the number of symmetries in the graph.



Index Positional Encodings

- The simplest possible PEs is to give an (arbitrary) ordering to the nodes, among $n!$ possible orderings.
- Theorem[1]: GCNs s.a. GINs[2] are provable more expressive than the *1*-WL test when considering all $n!$ node indices as PE node features.
- During training, orderings are uniformly sampled from the $n!$ possible choices in order for the network to learn to be independent to these arbitrary choices.



[1] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, 2019
[2] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- **Graph Positional Encodings**
  - Index Positional Encodings
  - **Structural Message Passing Networks**
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- GNNs for Sets

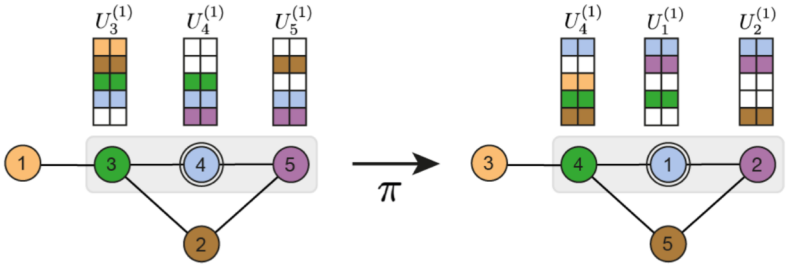
Structural Message-Passing Networks[1]

- Standard message-passing GNNs are
  - Index permutation equivariant (most fundamental GNN property/structural inductive bias), but
  - They have also limited expressive power representation (fail to distinguish non-isomorphic graphs beyond 1-WL[2], identify cycles[3] and basic graph geometric properties[4]).
- Limited expressivity can be overcome with unique node identifier[4].
  - Unique identifier breaks natural symmetries in graphs (s.a. node/edge isomorphism).
  - Natural choice is one-hot encoding of node index[5].
  - However, node indexing is arbitrary to $n!$ possible permutations, which makes impossible for the network to learn all possible permutations.
- Structural Message Passing (SMP) Networks[1] :
  - Idea : Equivariant processing of nodes by augmenting the node representation with a graph-size equivariant module, called local context matrix.

[1] Vignac, Loukas, Frossard, Building powerful and equivariant graph neural networks with message-passing, 2020
[2] Xu, Hu, Leskovec, Jegelka, How powerful are graph neural networks?, 2019
[3] Chen, Chen, Villar, Bruna, Can graph neural networks count substructures?, 2020
[4] Loukas, What graph neural networks cannot learn: depth vs width, 2019
[5] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, 2019

Structural Message-Passing Networks[1]

- SMP networks uses a local context matrix U that is equivariant, i.e. the rows of the matrix are simply permuted if the nodes are re-ordered.
  - Advantage : Allows initializing each node with a unique identifier (one-hot vector) without ambiguity and avoiding the $n!$ possible index permutations.



$$
\begin{aligned}
U_i^{\ell=0} &= 1_i \in \mathbb{R}^{n\times 1} \text{ (no input node features)} \\
&= \text{Concat}(1_i, x_i) \in \mathbb{R}^{n\times(c+1)} \text{ (with input node features)} \\
U_i^{\ell+1} &= f_{\text{update}}(U_i^{\ell}, \tilde{U}_i^{\ell}) \in \mathbb{R}^{n\times c_{\ell+1}} \\
\tilde{U}_i^{\ell} &= f_{\text{aggregation}}(f_{\text{message}}(\{U_i^{\ell}, U_j^{\ell}, e_{ij}\}_{j\in\mathcal{N}_i}) \in \mathbb{R}^{n\times c_{\ell+1}} \\
U^{\ell+1} &= \text{Concat}(U_1^{\ell+1}, ..., U_n^{\ell+1}) \in \mathbb{R}^{n\times n\times c_{\ell+1}}
\end{aligned}
$$

SMP memory/speed complexity is $O(n^2)$.
Standard MP is $O(n)$.

[1] Vignac, Loukas, Frossard, Building powerful and equivariant graph neural networks with message-passing, 2020

Structural Message-Passing Networks[1]

- Properties :
  - Ability to consider unique node identifiers (without ambiguities or combinatorial explosion).
  - Expressivity power : SMP networks can differentiate two non-isomorphic graphs.
  - Universality : SMP networks can represent any invariant function on graphs.
  - SMP are strictly more powerful than MP-GNNs.
  - SMP exploits graph sparsity, unlike E-GNNs[2].
  - Memory/speed complexity is $O(n^2)/O(n^2)$ for sparse graphs, which improves of 3-WL GNNs[3] with $O(n^2)/O(n^3)$.
    - Approximation : Supposing that graph diameter $> 2L$ (number of layers), that is no node can have twice the same identifier in its L-hop neighborhood, and using greedy node coloring (NP-complete) then complexities can be reduced to $O(n)$.
  - Numerical experiments demonstrate good performances to detect cycles of various lengths and graph topological properties (geodesic distance, node eccentricity, Laplacian features, connectivity, graph diameter, spectral radius), where MP-GNNs are known to fail.

[1] Vignac, Loukas, Frossard, Building powerful and equivariant graph neural networks with message-passing, 2020
[2] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019
[3] Maron, Ben-Hamu, Serviansky, Lipman, Provably powerful graph networks, 2019

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- **Graph Positional Encodings**
  - Index Positional Encodings
  - Structural Message Passing Networks
  - **Laplacian Positional Encodings**
- Link Prediction with Edge Representation
- GNNs for Sets

Laplacian Positional Encodings[1]

- Laplacian PE properties :
  - Hybrid positional and structural encodings, invariant by index permutation.
  - Unique and distance-sensitive : two nodes far away on a graph have large PEs distance, and inversely (v.s. two one-encoding vectors of different indices are equally distant).
- LapPEs have natural symmetries with the arbitrary sign of eigenvectors (after being normalized to have unit length).
  - The number of possible sign flips is 2^k, k being the number of eigenvectors.
  - In practice, we choose $k \ll n$, and therefore 2^k is much smaller $n!$ (the number of possible ordering of the nodes). Smaller sampling space than index PEs, and therefore smaller amount of ambiguities to be resolved by the network.
  - During the training, eigenvectors will be uniformly sampled at random between the 2^k possibilities.
- Lap PEs are graph generalizations of Transformers' PEs[2].

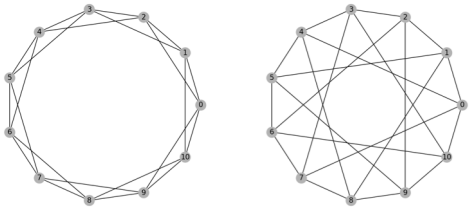
[1] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[2] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017

Laplacian Positional Encodings[1]

- Circular Skip Link (CSL) dataset[2] : Classify isomorphic graphs that differ by the skip link value.

Model	Test Acc±s.d.
MLP $O(n)$	22.567±6.089
GIN No-PE $O(n)$	10.000±0.000
GIN Lap-PE $O(n)$	99.333±1.333
GatedGCN No-PE $O(n)$	10.000±0.000
GatedGCN Lap-PE $O(n)$	**99.600±1.083**

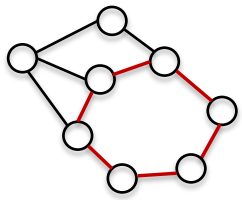
Table 1: Test accuracy on the CSL dataset. Results (higher is better) are averaged over 4 runs with 4 different seeds.



- CYCLES dataset[3] : Graphs that contain or not 6-cycles (binary classification task for graphs with cycle or not).

#Train samples	200	500	1000	5000
Model	Test Acc±s.d.			
SMP* $O(n^2)$	87.700±N.A.	97.400±N.A.	97.600±N.A.	99.500±N.A.
GIN index-PE* $O(n)$	65.800±N.A.	70.800±N.A.	80.600±N.A.	96.400±N.A.
GIN Lap-PE $O(n)$	76.097±9.561	94.295±0.648	96.790±0.269	99.380±0.112
GatedGCN Lap-PE $O(n)$	**94.850±0.221**	**96.712±0.156**	**98.778±0.086**	**99.740±0.027**

Table 2: Test accuracy on the CYCLES dataset. Results (higher is better) are averaged over 4 runs with 4 different seeds. *From Vignac et al., 2020.



[1] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[2] Murphy, Srinivasan, Rao, Ribeiro, Relational pooling for graph representations, 2019
[3] Vignac, Loukas, Frossard, Building powerful and equivariant graph neural networks with message-passing, 2020

Laplacian Positional Encodings[1]

- GraphTheory dataset[2] : Multi-task graph datasets with three node-level tasks (single-source shortest path, node eccentricity and Laplacian features) and three graph-level tasks (graph connectivity, diameter and spectral radius).
- Graphs are generated with Erdos-Rényi (20%), Barabási-Albert (20%), Grid(5%), Caveman (5%), Tree (15%), Ladder graphs (5%), Line graphs (5%), Star graphs (5%), Caterpillar graphs (10%), Lobster graphs (10%).

Model	Test						
	Average	Dist.	Ecc.	Lap.	Conn.	Diam.	Rad.
GIN* $O(n)$	-1.99±N.A	-2.00±N.A	-1.90±N.A	-1.60±N.A	-1.61±N.A	-2.17±N.A	-2.66±N.A
PNA* $O(n)$	-3.13±N.A	-2.89±N.A	-2.89±N.A	-3.77±N.A	-2.61±N.A	-3.04±N.A	-3.57±N.A
SMP† $O(n^2)$	**-3.59±N.A**	**-3.59±N.A**	**-3.67±N.A**	**-4.27±N.A**	**-2.97±N.A**	**-3.58±N.A**	-3.46±N.A
GatedGCN No-PE $O(n)$	-2.83±0.13	-2.76±0.17	-2.36±0.12	-3.92±0.15	-2.65±0.11	-3.35±0.16	-4.31±0.08
GatedGCN Lap-PE $O(n)$	-3.18±0.11	-3.23±0.08	-3.35±0.08	-4.03±0.21	-2.60±0.12	-3.57±0.05	**-4.32±0.13**

Table 3: Test MSE on the GraphTheory dataset. Average denotes the combined average of all the tasks. Results (lower is better) are averaged over 4 runs with 4 different seeds. *From Corso et al., 2020 and †from Vignac et al., 2020.

[1] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020
[2] Corso, Cavalleri, Beaini, Lio, and Velickovic, Principal neighbourhood aggregation for graph nets, 2020

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- **Link Prediction with Edge Representation**
- GNNs for Sets

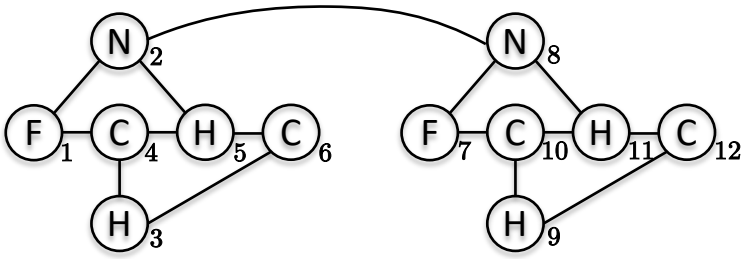
Link Prediction Task

- GCNs can fail the link prediction task.
  - Apply any GCN to this molecule composed of two identical compounds :

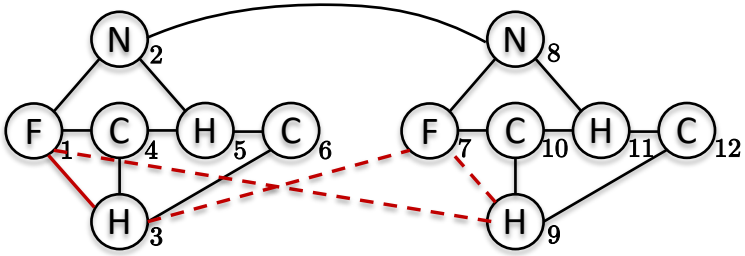
We have

$h_1 = h_7 \in \mathbb{R}^d$

$h_3 = h_9 \in \mathbb{R}^d$

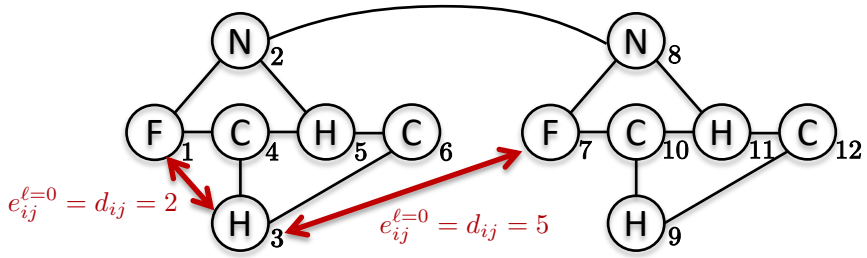


  - Perform link prediction (by transfer learning) :
    - Suppose there exist a bond $(\mathrm{F}_{(\text{item 1})}, \mathrm{H}_{(\text{item 3})})$,
    - Then can we predict the link $(\mathrm{F}_{(\text{item 7})}, \mathrm{H}_{(\text{item 9})})$? Yes, but the network will also predict (wrong) links between $(\mathrm{F}_{(\text{item 7})}, \mathrm{H}_{(\text{item 3})})$ and between $(\mathrm{F}_{(\text{item 1})}, \mathrm{H}_{(\text{item 9})})$.



Link Prediction Task

- How to design expressive GCNs for the link prediction task ?
- Theorem[1] :
  - Link prediction is maximally expressive with a joint representation of nodes (intuitively this encodes the similarity/distance between all pairs of nodes).
  - This requires $O(n^2)$ edge representations.

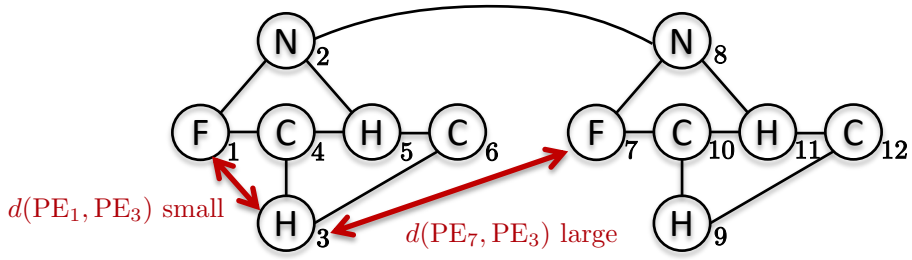


Input edge representation is
$e_{ij}^{l=0} = d_{ij}$ (e.g. shortest distances).

[1] Srinivasan, Ribeiro, On the Equivalence between Positional Node Embeddings and Structural Graph Representations, 2019

Link Prediction Task

- Alternate solution[1] :
  - Represent links as joint representations of nodes.
  - Add node positional encodings that are unique and distance-sensitive to differentiate links, like Laplacian PEs.
  - Complexity is O(E), E is the number of edges.



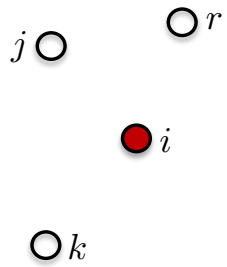
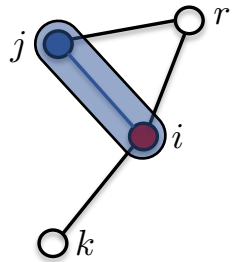
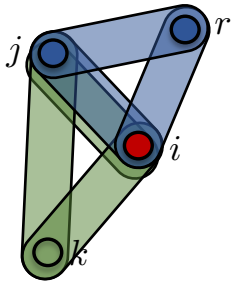
[1] Dwivedi, Joshi, Laurent, Bengio, Bresson, Benchmarking graph neural networks, 2020

Outline

- Weisfeiler-Lehman GNNs
  - Graph Isomorphism Networks
  - Principal Neighbourhood Aggregation
  - Equivariant GNNs
  - 3-WL/Ring GNNs
  - Sparse WL-GNNs
  - Low-Rank Attention GNNs
  - Graph Substructure Networks
- Expressivity Power and Universal Approximator
  - Expressivity with WL
  - Graph Universal Approximator
  - Limitations of Expressivity
- Graph Positional Encodings
  - Index Positional Encodings
  - Structural Message Passing Networks
  - Laplacian Positional Encodings
- Link Prediction with Edge Representation
- **GNNs for Sets**

Beyond Graphs

- Tensors invariant by permutation :

Sets	Graphs	Tensors
First-order tensors (Unordered) vectors *1*-tuple of nodes	Second-order tensors Graphs *2*-tuple of nodes	Order-m tensors Hyper-graphs m-tuple of nodes
		
No edge	Edges $= \{i,j\},\{i,r\},\{i,k\}$	Hyper-edges $=\{i,j,r\},$ $\{i,k,r\},\{i,j,k\}$

- Equivariant GNNs[1] can be run on any tensor of order-m, with memory/speed complexities $O(n^{\max(m,m')})$.

$$f_W : \mathbb{R}^{n^m} \to \mathbb{R}^{n^{m'}}$$
$$g_W : \mathbb{R}^{n^m} \to \mathbb{R}^K, K \geq 1$$

[1] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Equivariant/Invariant Linear Layer for Sets

- For rank-1 tensors/vectors, the equivariant linear transformation $f_W : \mathrm{R}^n$ to R^n is fully characterized by bell(2)=2 equivariant functions, which are I and 11^T-I. The parametrized equivariant linear transformation is[1,2,3] :

$$f_W(x) = \left[w_1 \mathrm{I} + w_2 \left(11^T - \mathrm{I}\right)\right] x \in \mathbb{R}^n$$

$$f_W(x)_i = w_1 x_i + w_2 \sum_{j \neq i} x_j$$

$$f_W\left(P \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = f_W\left(\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right) = f_W\left(\begin{bmatrix} x_3 \\ x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} w_1 x_3 + w_2(x_1 + x_2) \\ w_1 x_1 + w_2(x_2 + x_3) \\ w_1 x_2 + w_2(x_1 + x_3) \end{bmatrix} = P f_W\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}\right)$$

- The invariant linear transformation $g_W : \mathrm{R}^n$ to R^K is is fully characterized by bell(1)=1 equivariant function, which is $1_{\mathrm{K\ x\ n}}$. The parametrized invariant linear transformation is

$$g_W(x) = w_1 1_{K \times n} x \in \mathbb{R}^K, K \geq 1$$

- Linear equivariant functions for sets have limited expressivity (only 2 parameters to learn per layer), and thus cannot be performant for NLP tasks (text is rank-1 tensor).

[1] Zaheer, Kottur, Ravanbakhsh, Poczos, Salakhutdinov, Smola, Deep sets, 2017
[2] Qi, Su, Mo, Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017
[3] Maron, Ben-Hamu, Shamir, Lipman, Invariant and equivariant graph networks, 2019

Equivariant/Invariant Linear Layer for Sets

- How to improve the expressivity of linear equivariant functions for sets ?
  - Consider non-linear equivariant functions.
    - Transformers[1] use (non-linear) attention functions to extract word representations that adapt to the context.
    - Very successful in NLP

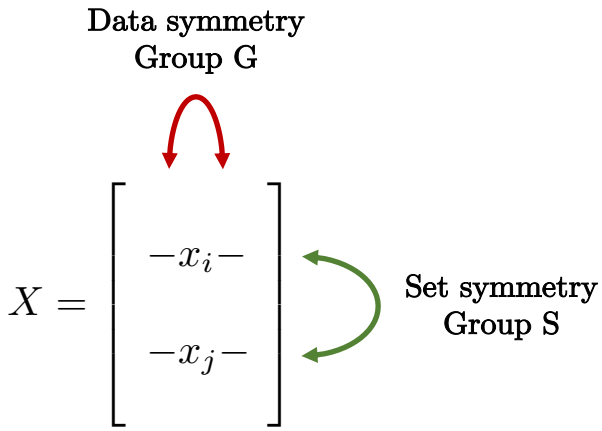
$$h_i^{\ell+1} = f_W(A) = \sum_{j \in V} \frac{\exp(W_1 h_j^\ell)}{\sum_{k \in V} \exp(W_2 h_k^\ell)} W_3 h_j^\ell$$

  - Consider additional symmetry/invariance[2,3].

[1] Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin, Attention is all you need, 2017
[2] Zaheer, Kottur, Ravanbakhsh, Poczos, Salakhutdinov, Smola, Deep sets, 2017
[3] Maron, Litany, Chechik, Fetaya, On Learning Sets of Symmetric Elements, 2020

On Learning Sets of Symmetric Elements[1]

- Neural networks for sets of symmetric data.
  - Set symmetry : order invariance
  - Data symmetry : translation invariance for example
- A set of data can be represented as a matrix $X \in R^{n \times d}$



- DeepSet[2] : Neural network for sets of arbitrary data (order invariance only)

$$f^S_W(X)_i = f^S_{W_1}(x_i) + f^S_{W_2}(\sum_{j \neq i} x_j)$$

$$f^S_{W_1}(x) = w_1 x, \;\; f^S_{W_2}(x) = w_2 x$$

- ConvNets[3] : Neural network for symmetric data (translation invariance only)

$$f^G_W(x_i) = W * x_i$$

[1] Maron, Litany, Chechik, Fetaya, On Learning Sets of Symmetric Elements, 2020

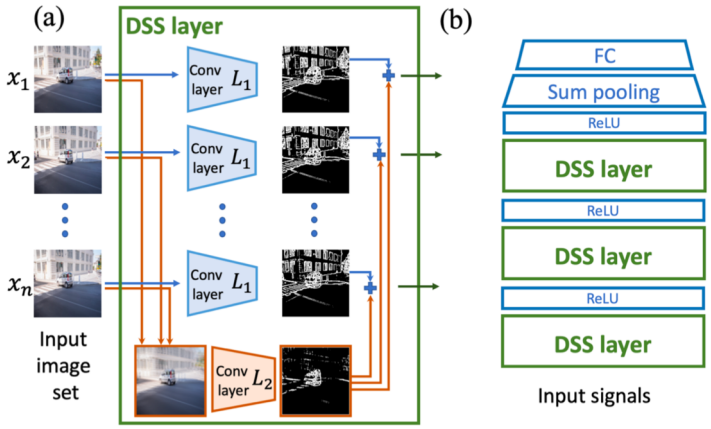
On Learning Sets of Symmetric Elements[1]

- Deep Sets for Symmetric elements (DSS)[1] : Combining symmetries from DeepSets and ConvNets into a single framework.
- Any linear equivariant layer for the symmetry group S × G is of the form :

$$f_W^{S\times G}(X)_i = f_{W_1}^G(x_i) + f_{W_2}^G(\sum_{j\neq i} x_j),$$

$$f_W^G(x) = W * x$$

- Expressivity power : If G-equivariant networks are universal, then S × G-equivariant networks are also universal, that is it can represent any function with the S × G symmetry.
- Applications : 1D-signals, 3D cloud points and set of images.



[1] Maron, Litany, Chechik, Fetaya, On Learning Sets of Symmetric Elements, 2020

Questions?